updated, so that the update is available on all deployed systems at the same time. The manner in which this is accomplished in preferred embodiments will now be described with reference to Fig. 11, where encircled numbers correspond to the steps listed below.

1. The update process preferably starts when the deployment provider updates the web service on the origin server 290. This may be done by shutting down the web service, updating the service's executable code and meta information, and then starting the service again.

2. The deployment provider 280 will issue an update request from the origin server 290 to the deployment node 260. The deployment provider is preferably the only participant in this scheme that is allowed to issue this type of request.

3. The deployment node will verify that the web service has been deployed, by searching for it in its private registry 270. The private registry contains a reference to the web service that is deployed on the origin server 290, as well as all of the dynamically deployed web services.

4. If the web service is found in the private registry, then the update request will be processed by the deployment node. The deployment node is responsible for completing the update request from the deployment provider. If the web service is not found in the private registry, then an error message is preferably sent back to the deployment provider (not shown in Fig. 11).

5. The deployment node starts the update process by sending an unpublish request to the public registry 220. After this request is processed, service requesters 210 will not be able to find the web service, but all of the web services will still be running.

6. After the unpublish request is completed, the deployment node will obtain a list from its registry 270 of the deployment facilitators 230 where the web service was deployed.

7. For each web service on the list, the DNS server 250 is updated so that all requests are sent to the deployment node. While the update is being processed, the deployment node will preferably return an error message to any service requesters that try to access the service.

8. After the DNS server is updated, the deployment node sends an update request to each deployment facilitator in the list obtained in step 5. Preferably, this update request is formatted in a similar manner and uses a similar syntax as the deployment request which is sent in stage 8 of the deployment process (see Fig. 4), but differs in that it conveys that this is an update request.

9. When the deployment facilitator receives an update request, it will shut down the web service and then send a deployment request to the origin server to obtain the updated web service. In preferred embodiments, this (re)deployment request uses the same format and syntax which has previously been described with reference to stage 9 (see Fig. 4) of the deployment process.

10. After the deployment facilitator receives the updated web service package from the deployment provider 280, it will deploy the web service code and meta information in the run-time environment and then start the service.

11. When the deployment node has completed the update request from the deployment provider, it will send a request to the DNS server to re-activate the original DNS entries for the deployed web services and will publish the updated service description to the public registry 220.

12. After the service description is published, a service requestor can find the updated service description and use any of the updated web services. Service requests will be re-routed to the deployed web services by the DNS server, or will be routed to the deployment node and forwarded to the web service that is running on the origin server, in the manner which has previously been described for stages 5 and 6 of the deployment process. (Refer to Fig. 4 for a depiction of these flows, which have not been shown in Fig. 11.)

13. The update process is preferably completed when the deployment node sends a response to the update request to the deployment provider (not shown in Fig. 11).

Thus, it can be seen that use of either or both of the optional enhancements provides advantages in addition to the dynamic deployment technique which has been described.

As will be appreciated by one of skill in the art, embodiments of the present invention may

be provided as methods, systems, or computer program products. Accordingly, the present invention may take the form of an entirely hardware embodiment, an entirely software embodiment or an embodiment combining software and hardware aspects. Furthermore, the present invention may take the form of a computer program product which is embodied on one or more computer-usable storage media (including, but not limited to, disk storage, CD-ROM, optical storage, and so forth) having computer-usable program code embodied therein.

The present invention has been described with reference to flow diagrams and/or block diagrams of methods, apparatus (systems) and computer program products according to embodiments of the invention. It will be understood that each flow and/or block of the flow diagrams and/or block diagrams, and combinations of flows and/or blocks in the flow diagrams and/or block diagrams, can be implemented by computer program instructions. These computer program instructions may be provided to a processor of a general purpose computer, special purpose computer, embedded processor or other programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer or other programmable data processing apparatus, create means for implementing the functions specified in the flow diagram flow or flows and/or block diagram block or blocks.

These computer program instructions may also be stored in a computer-readable memory that can direct a computer or other programmable data processing apparatus to function in a particular manner, such that the instructions stored in the computer-readable memory produce an article of manufacture including instruction means which implement the function specified in the

flow diagram flow or flows and/or block diagram block or blocks.

The computer program instructions may also be loaded onto a computer or other programmable data processing apparatus to cause a series of operational steps to be performed on the computer or other programmable apparatus to produce a computer implemented process such that the instructions which execute on the computer or other programmable apparatus provide steps for implementing the functions specified in the flow diagram flow or flows and/or block diagram block or blocks.

While the preferred embodiments of the present invention have been described, additional variations and modifications in those embodiments may occur to those skilled in the art once they learn of the basic inventive concepts. Therefore, it is intended that the appended claims shall be construed to include both the preferred embodiment and all such variations and modifications as fall within the spirit and scope of the invention.

What is claimed is:

1. A method of dynamically undeploying services in a computing network, comprising steps of:

receiving an undeployment trigger for a selected service;

determining one or more network locations where the selected service is deployed; and

effecting a dynamic undeployment by programmatically removing the selected service from one or more selected ones of the network locations.

2. The method according to Claim 1, further comprising the steps of:

receiving client requests for the selected service; and

continuing to serve the received requests from the network locations other than the one or more selected ones from which the selected service was programmatically removed.

3. The method according to Claim 1, wherein the undeployment trigger is based upon usage of the selected service at the network locations.

4. The method according to Claim 1, wherein the undeployment trigger is based upon network load at the network locations.

5. The method according to Claim 1, wherein the undeployment trigger is an undeployment request issued by an origin server from which the selected service was initially deployed.

6. The method according to Claim 3, further comprising comparing the usage of the selected service to a predetermined threshold, and sending the undeployment trigger when the usage falls below the predetermined threshold.

7. The method according to Claim 6, wherein a value of the predetermined threshold may be modified over time.

8. The method according to Claim 6, wherein a value of the predetermined threshold applies to a plurality of deployed services.

9. The method according to Claim 6, wherein the predetermined threshold applies individually to the selected service.

10. The method according to Claim 6, wherein a value of the predetermined threshold applies to all of the network locations.

11. The method according to Claim 6, wherein a value of the predetermined threshold applies to the one or more selected ones of the network locations.

12. The method according to Claim 6, wherein a value of the predetermined threshold is initially set when the selected service is deployed.

13. The method according to Claim 3, wherein the usage is an average number of client requests for the selected service within a predetermined time interval.

14. The method according to Claim 6, further comprising the step of obtaining the usage at periodic intervals for use in the comparing step.

15. The method according to Claim 14, wherein the obtaining step obtains the usage from all of the network locations.

16. The method according to Claim 15, wherein the obtaining step obtains the usage from representative ones of the network locations.

17. The method according to Claim 15, wherein the programmatically removing occurs at a particular one of the network locations, and wherein the obtaining step obtains the usage from the particular one.

18. The method according to Claim 1, further comprising the steps of:

monitoring a load on the computing network; and

triggering the dynamic undeployment when the monitored load meets a predetermined threshold.

19. The method according to Claim 1, wherein the programmatically removing step further

comprises the step of issuing an undeployment request for the selected service to the one or more selected ones.

20. The method according to Claim 19, further comprising steps of:

receiving the undeployment request at a particular one of the network locations, the particular one being the selected one of the network locations from which the selected service is being dynamically undeployed; and

shutting down the selected service at the particular one, responsive to the receiving step, and removing executed code which implements the selected service from a run-time environment of the particular one.

21. The method according to Claim 20, further comprising the step of making the selected service unlocatable from a routing system.

22. The method according to Claim 5, further comprising steps of:

sending the undeployment request to all of the network locations;

shutting down the selected service at the network locations, responsive to the receiving step, and removing executed code which implements the selected service from a run-time environment of each network location;

shutting down the selected service at the origin server; and, responsive to the receiving step, and removing executed code which implements the selected service from a run-time environment of each network location; and

making the selected service unlocatable in the computing network.

23. A system for dynamically undeploying services in a computing network, comprising:

means for receiving an undeployment trigger for a selected service;

means for determining one or more network locations where the selected service is deployed; and

means for effecting a dynamic undeployment by programmatically removing the selected service from one or more selected ones of the network locations.

24. A computer program product for dynamically undeploying services in a computing network, the computer program product embodied on one or more computer-readable media and comprising:

computer-readable program code means for receiving an undeployment trigger for a selected service;

computer-readable program code means for determining one or more network locations where the selected service is deployed; and

computer-readable program code means for effecting a dynamic undeployment by programmatically removing the selected service from one or more selected ones of the network locations.